UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Introduction to Number Theory and Modular Arithmetic

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Cybersecurity, 2022/2023

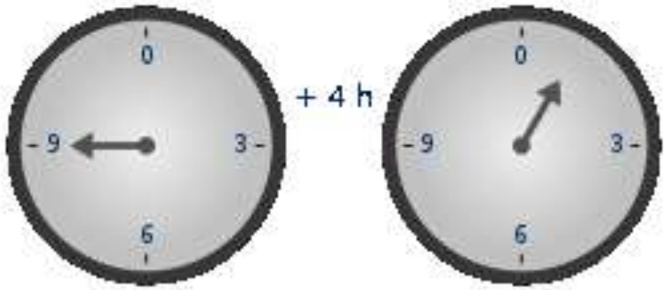
http://netsec.unipr.it/veltri

Modular Arithmetic

- Define **modulo operator** $a \bmod n$ to be remainder when a is divided by n
  - **$a = qn + r$, where: q = quotient = $\lfloor a/n \rfloor$ and $0 \le r \le n-1$**
  - **$r = a \bmod n$**
- *n* is called the **modulus**
- The result *r* of the modulo operation is called the **residue** of *a* mod *n*
- Given two integers a and b, they are said to be "**congruent** modulo *n*" if, when divided by n, a and b have same remainder
  - **it means that ($a \bmod n$) = ($b \bmod n$)**
  - **Use the term congruence for: $a \equiv b \pmod n$**
- Examples
  - **$100 \equiv 34 \pmod{11}$**
  - **$-12 \bmod 7 = -5 \bmod 7 = 2 \bmod 7 = 9 \bmod 7$**
- Note:
  - **if $a \equiv 0 \pmod n$, then $n|a$**

Modular Arithmetic (cont.)

- All operations have a result between 0 and *n-1*
- It is "clock arithmetic"
  - **uses a finite number of values, and loops back from either end**
  - **do addition & multiplication and modulo reduce**



Modular Arithmetic (cont.)

- Properties of congruence:
  - **$a \equiv b \pmod{n} \quad \Leftrightarrow \quad b \equiv a \pmod{n}$**
  - **$a \equiv b \pmod{n} \quad \Leftrightarrow \quad a-b \equiv 0 \pmod{n} \quad \Leftrightarrow \quad n|(a-b)$**
  - **$\forall k\,,\ a \equiv a + kn \pmod{n}$**
  - **$a \equiv b \pmod{n}$ and $b \equiv c \pmod{n} \quad \Rightarrow \quad a \equiv c \pmod{n}$**
- Properties of addition, subtraction and product:
  - **a <u>op</u> b ≡ (a mod n) <u>op</u> (b mod n) (mod n)**
  - **i.e. (a <u>op</u> b) mod n = ((a mod n) <u>op</u> (b mod n)) mod n**
    - with: <u>op</u> = +, -, *
  - **result: can do reduction at any point, i.e.**
    - (a+b) mod n = ((a mod n) + (b mod n)) mod n
  - **the property with the product leads also the following result:**
    - $a^k \equiv (a \bmod n)^k \pmod{n}$, i.e. $a^k \bmod n = (a \bmod n)^k \bmod n$
- Example of use:
  - **10^3 mod 7 = 1000 mod 7 = 142*7 + 6 = 6**
    - however it is easier to compute as:
      10^3 mod 7 = (10 mod 7) ^3 mod 7 = 3^3 mod 7 = 27 mod 7 = 6

Z_n

- Z_n is defined as the set of all integers ≥0 and <n
  - **Z_n = {0, 1, … , n-1}**
  - **called set of residues (mod *n*), or set of remainders (mod *n*), or set of residue classes (mod *n*)**
- Z_n forms a commutative ring for addition with a multiplicative identity element

Property	Expression
Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x + w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse ($-w$)	For each $w \in Z_n$, there exists a z such that $w + z \equiv 0 \bmod n$

Example – Arithmetic modulo 8

Addition modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

Multiplication modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

Additive and multiplicative inverses modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

Divisors

- Say a non-zero number b **divides** a if for some m have $a=mb$ (a,b,m all integers)
  - **that is b divides into a with no remainder**
  - **denote this $b|a$**
  - **and say that b is a divisor of a**
- Example
  - **all of 1,2,3,4,6,8,12,24 divide 24**

Prime Numbers

- Prime numbers only have divisors of 1 and self
  - **they cannot be written as a product of other numbers**
  - **note: 1 is prime, but is generally not of interest**
- e.g. 2,3,5,7 are prime, 4,6,8,9,10 are not
- Prime numbers are central to number theory
- List of prime numbers less than 200 is:

  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

Prime Factorization

- To **factor** a number *n* is to write it as a product of other numbers:
  `n=a × b × c`
- Note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- The **prime factorization** of a number *a* is when it's written as a product of primes
  - **eg.** $91=7\times13$; $3600=2^4\times3^2\times5^2$

$$a = \prod_{p \in P} p^{a_p}$$

Relatively Prime Numbers

- Two numbers `a,b` are **relatively prime** if have **no common divisors** apart from 1

- Example
  - 8 and 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor

Greatest Common Divisor (GCD)

- A common problem in number theory
- GCD of *a* and *b*, that is *GCD(a,b)*, is the largest number that divides both *a* and *b*
  - eg GCD(60,24) = 12
- The greatest common divisor can be determined by comparing their prime factorizations and using least powers
  - Example
    - $300=2^1\times3^1\times5^2$ $18=2^1\times3^2$ hence $GCD(18,300)=2^1\times3^1\times5^0=6$
- Often want **no common factors** (except 1)
  - hence numbers are relatively prime
  - e.g. GCD(8,15) = 1
    - 8 & 15 are relatively prime

Euclid's GCD Algorithm

- An efficient way to find the *GCD(a,b)*
- Uses theorem that:
  - **`GCD(a,b) = GCD(b, a mod b)`**
  - **dim**
    - if $d|a,b$ (d divides a and b), then $a=h*d$ and $b=k*d$ where h,k are the quotients, then $a \bmod b = r_a = a - q_a b = hd - q_a kd = (h-q_a k)d \Rightarrow d|(a \bmod b)$
    - moreover, starting form $a = r_a + q_a b$, if $d|b$ and $d|r_a$, with $r_a=s*d$, then $a= r_a + q_a b = (s+q_a k)d \Rightarrow d|a$
  - **so, common divisors of a,b are also common divisor of b and r_a**

Euclid's GCD Algorithm (cont.)

- If we use it several times:
  - $\mathrm{GCD}(a,b) = \mathrm{GCD}(b, a \bmod b) = \mathrm{GCD}(b, r_1) = \mathrm{GCD}(r_1, b \bmod r_1) = \mathrm{GCD}(r_1, r_2) = \mathrm{GCD}(r_2, r_1 \bmod r_2) = \mathrm{GCD}(r_2, r_3) = \ldots = \mathrm{GCD}(r_n, 0) = r_n$

    where:
    - $r_1 = a \bmod b$
    - $r_2 = b \bmod r_1$
    - $r_3 = r_1 \bmod r_2$
    - ...
  - $r_k = r_{k-2} \bmod r_{k-1}$
  - **the formula is valid also for r_1 and r_2 if we define:**
    - $r_0 = b$, $r_{-1} = a$
  - **note: at each step, $r_k < r_{k-1}$**
  - **calculate r_k using r_{k-1} and r_{k-2}, until $r_{n+1} = r_{n-1} \bmod r_n$ is equal to 0, then**
  - $\mathrm{GCD}(a,b) = \mathrm{GCD}(r_n, 0) = r_n$

Euclid's GCD Algorithm (cont.)

- **Euclid's Algorithm** to compute *GCD(a,b)*:

```
A←a, B ← b
while B>0 {
    R ← A mod B
    A ← B
    B ← R
 }
return A
```

Example GCD(1970,1066)

gcd(1970, 1066)	1970 = 1 x 1066 + 904
gcd(1066, 904)	1066 = 1 x 904 + 162
gcd(904, 162)	904 = 5 x 162 + 94
gcd(162, 94)	162 = 1 x 94 + 68
gcd(94, 68)	94 = 1 x 68 + 26
gcd(68, 26)	68 = 2 x 26 + 16
gcd(26, 16)	26 = 1 x 16 + 10
gcd(16, 10)	16 = 1 x 10 + 6
gcd(10, 6)	10 = 1 x 6 + 4
gcd(6, 4)	6 = 1 x 4 + 2
gcd(4, 2)	4 = 2 x 2 + 0
gcd(2, 0)	

Multiplicative inverse (modulo *n*)

- The multiplicative inverse of a number x is the number we multiply x by to get 1
  - **with real numbers this is just 1/x**
  - **the multiplicative inverse of *m mod n* is *u : u*m = 1 (mod n)***
    - *u*m* differs from 1 by a multiple of *n*, or *u*m + v*n = 1*
- The Extended Euclid's Algorithm can be used to find the multiplicative inverse (if it exists)
  - **solving the problem:**
    - Find u,v | u*m + v*n = 1
- Theorem: a number *m* has a multiplicative inverse m^{-1} (mod *n*) if and only if *m* and *n* are relatively prime (co-prime)
  - **that is, if and only if *gcd(m,n)=1***

Extended Euclid's Algorithm

- Not only calculates the $d=gcd(a,b)$, but also two integer x and y (of opposite sign) such that:

$$x\,a + y\,b = d = \gcd(a,b)$$

- From Euclid's Algorithm:
  - $r_k = r_{k-2} \bmod r_{k-1} = r_{k-2} - q_k\, r_{k-1}$, where: $q_k = \lfloor r_{k-2} / r_{k-1} \rfloor$

  with:
  - $r_{-1} = a$
  - $r_0 = b$

- Writing the previous equation as function of *a* and *b*:
  - $r_1 = a \bmod b = a - q_1 b =^{(def)} x_1 a + y_1 b$
  - $r_2 = b \bmod r_1 = b - q_2 r_1 = b - q_2(x_1 a + y_1 b) = -q_2 x_1 a + (1-q_2 y_1) b =^{(def)} x_2 a + y_2 b$
  - $r_3 = r_1 \bmod r_2 = r_1 - q_3 r_2 = (x_1 - q_3 x_2) a + (y_1 - q_3 y_2) b =^{(def)} x_3 a + y_3 b$
  - …
  - $r_k = r_{k-2} \bmod r_{k-1} =^{(def)} x_k a + y_k b$

  with:
  - $x_k = x_{k-2} - q_k\, x_{k-1}$, with: $x_{-1}=1$, $x_0=0$
  - $y_k = y_{k-2} - q_k\, y_{k-1}$, with: $y_{-1}=0$, $y_0=1$

Extended Euclid's Algorithm (cont.)

- Algorithm:
  - **set**
    - $r_{-1} = a \;,\; x_{-1}=1 \;,\; y_{-1}=0$
    - $r_0 = b \;,\; x_0=0 \;,\; y_0=1$
  - **compute**
    - $r_k = r_{k-2} \bmod r_{k-1} = r_{k-2} - q_k\, r_{k-1}$, where $q_k = \lfloor r_{k-2} / r_{k-1} \rfloor$
    - $x_k = x_{k-2} - q_k\, x_{k-1}$
    - $y_k = y_{k-2} - q_k\, y_{k-1}$
  - **until**
    - $r_{n+1} = 0$
  - **then, it is**
    - $d = r_n = x_n\, a + y_n\, b$
  - **that gives both the GCD *d* and the values *x* and *y* such that**
    - $d = x\, a + y\, b$

Computation of the multiplicative inverse (modulo *n*)

- If the parameter *a* is a modulus *n*, and the parameter *b* is an integer *m* such that *GCD(m,n)=1*, then the algorithm gives the coefficients *x* and *y* such that

  x*n + y*m = 1

  y*m = 1 - xn

  that can be written as

  y*m = 1+kn

  that says that *y* is the multiplicative inverse of *m* modulo *n*

- Note
  - **If the value of the coefficient *y* is negative (-*w*), the residue modulo *n* (between 0 and n-1) multiplicative inverse of *m* can be simply obtained as**

    y + n = n - w

Extended Euclid's Algorithm - Example 1

k	q_k r_k	x_k	y_k	
-1	43	1	0	
0	35	0	1	
1	$43 = 1 \cdot 35 + 8$	1	-1	$8 = 1 \cdot 43 + (-1) \cdot 35$
2	$35 = 4 \cdot 8 + 3$	-4	5	$3 = (-4) \cdot 43 + 5 \cdot 35$
3	$8 = 2 \cdot 3 + 2$	9	-11	$2 = 9 \cdot 43 + (-11) \cdot 35$
4	$3 = 1 \cdot 2 + 1$	-13	16	$1 = (-13) \cdot 43 + 16 \cdot 35$
5	$2 = 2 \cdot 1 + 0$			

- Algorithm start with k=1
- At each step the new coefficients r_k, q_k are calculated:
  - **$q_k = \lfloor r_{k-2} / r_{k-1} \rfloor$**
  - **$r_k = r_{k-2} \bmod r_{k-1}$**
  - **that is: $r_k = r_{k-2} - q_k\, r_{k-1}$**
- From r_k, e q_k , the new values of x_k e y_k are calculated:
  - **$x_k = x_{k-2} - q_k\, x_{k-1}$**
  - **$y_k = y_{k-2} - q_k\, y_{k-1}$**

Extended Euclid's Algorithm - Example 2

- Solve 1759 x + 550 y = gcd(1759, 550)

i	q_i	r_i	x_i	y_i
-1		1759	1	0
0		550	0	1
1	3	109	1	-3
2	5	5	-5	16
3	21	4	106	-339
4	1	1	-111	355

- Result:
  - ***d* = 1; *x* = - 111; *y* = 355**
  - **(355)*550 + (-111)*1759 = 1**
    - (355)*550 = 1 + k 1759
    - 355 is the multiplicative inverse of 550 (mod 1759)

Galois Fields GF(p)

- If p is prime, all integers x with $x<p$ are relatively prime with p
- $Z_p = \{0,1, \dots , p-1\}$ with arithmetic operations modulo prime p form a finite field (aka know as Galois Field)
  - **since have multiplicative inverses**
- GF(p) = Galois field of order p
- Arithmetic is “well-behaved” and can do addition, subtraction, multiplication, and division without leaving the field GF(p)

Example – Arithmetic in GF(7)

Addition modulo 7

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Multiplication modulo 7

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Additive and multiplicative inverses modulo 7

w	$-w$	w^{-1}
0	0	—
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

Euler Totient Function ø(n)

- When doing arithmetic modulo n
- **Complete set of residues** is: `0..n-1`
- **Reduced set of residues** is those numbers (residues) which are relatively prime to n
  - **eg for n=10,**
  - **complete set of residues is {0,1,2,3,4,5,6,7,8,9}**
  - **reduced set of residues is {1,3,7,9}**
- Number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

Euler Totient Function $\varnothing(n)$

- To compute ø(n) need to count number of elements to be excluded
- In general need prime factorization, but
  - **for p (p prime)** `ø(p) = p-1`
  - **for p.q (p,q prime)** `ø(p.q) = (p-1)(q-1)`
- Examples
  - `ø(37) = 36`
  - `ø(21) = (3–1)×(7–1) = 2×6 = 12`

Euler's Theorem

- $a^{\varnothing(n)} \bmod n = 1$
  - where gcd(a,n)=1
- e.g.
  - $a=3; n=10;\ \varnothing(10)=4;$
  - hence $3^4 = 81 = 1 \bmod 10$
  - $a=2; n=11;\ \varnothing(11)=10;$
  - hence $2^{10} = 1024 = 1 \bmod 11$
- It generalizes the Fermat's (Little) Theorem that says:
  - $a^{p-1} \bmod p = 1$
    - where p is prime and gcd(a,p)=1
- Corollary from Euler's Theorem
  - $a^{k\varnothing(n)+1} \bmod n = a$

Primitive Roots

- Consider the equation $a^m \bmod n = 1$
  - **If gcd(a,n)=1 then *m* does exist**
    - Euler's theorem gives m=$\varnothing(n)$, but may be smaller
  - **note: once powers reach *m*, cycle will repeat: $a^{m+k} = a^m \cdot a^k = a^k$**
- The smallest m such that $a^m=1$ is called multiplicative order of *a* modulo n
- A number g is a primitive root modulo n if every number coprime to n is congruent to a power of g modulo n
  - ***g* is also called "generator of the multiplicative group of integers modulo *n*" (the reduced set of residues)**
    - $\forall$ *b* , gcd(b,n)=1, $\exists$ k : $g^k \equiv b \pmod n$
      - such *k* is called the index or discrete logarithm of *b* to the base *g* modulo *n*
  - **if n=p is prime, then successive powers of g "generate" the group mod p**
- if the multiplicative order of a modulo n is $m=\varnothing(n)$ then a is a primitive root (modulo n)

Discrete Logarithms

- The inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo *p*
- That is to find *x* where $a^x = b \pmod{p}$
- Written as $x = \log_a b \bmod p = dlog_{a,p}(b)$
- If *a* is a primitive root and *p* is prime then always exists, otherwise may not
  - **Examples**
    - $x = \log_3 5 \pmod{13}$ has no answer
    - $x = \log_2 5 \pmod{13} = 9$, by trying successive powers
  - **Note, in case of modulus *n* not prime, it exists if:**
    - *a is* primitive root, *b* is co-prime with *n*
- Whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Primality Testing

- Often need to find large prime numbers
- Traditionally **check by** using **trial division**
  - **i.e. divide by all numbers (primes) in turn less than the square root of the number**
  - **only works for small numbers**
- Alternatively can use statistical primality tests based on properties of primes
  - **for which all primes numbers satisfy property**
  - **but some composite numbers, called pseudo-primes, also satisfy the property**

Miller Rabin Algorithm

- A test based on Fermat's Theorem
- Algorithm is:

TEST (n) is:

1. Find integers k, q, with $k > 0$, q odd, so that $(n-1)=2^k q$

2. Select a random integer a, $1<a<n-1$

3. if $a^q \bmod n = 1$ then **return ("maybe prime");**

4. for $j = 0$ to $k - 1$ do

    5. if $(a^{2^j q} \bmod n = n-1)$

        then **return("maybe prime")**

6. return ("composite")

Miller Rabin Algorithm (cont.)

- if Miller-Rabin returns “composite” the number is definitely not prime
- Otherwise is a prime or a pseudo-prime
- Chance it detects a pseudo-prime is < ¼
- Hence if repeat test with different random *a* then chance *n* is prime after *t* tests is:
  - **Pr(*n* prime after *t* tests) = $1\text{-}4^{-t}$**
  - **eg. for t=10 this probability is > 0.99999**